

OpenInfobutton Development Environment Setup Guide

Table of Contents

[1 Software Requirements 1](#_Toc365974898)

[2 Eclipse Plugins 1](#_Toc365974899)

[2.1 EGit 2](#_Toc365974900)

[2.2 M2E 8](#_Toc365974901)

[3 Retrieval from GitHub 16](#_Toc365974902)

[4 Building 17](#_Toc365974903)

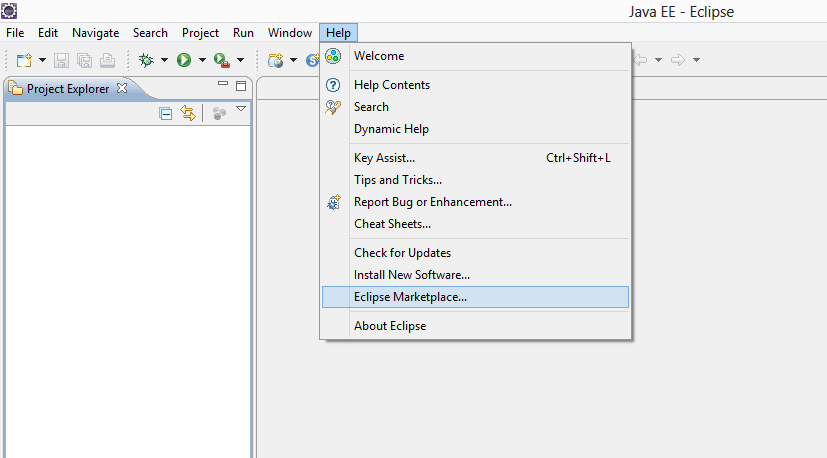
[5 Running 18](#_Toc365974904)

# Software Requirements

* Eclipse 3.7 Indigo (or better) for Java EE
* Java 6 JDK or better
* Tomcat 6
* Maven 2.2
* JAVA\_HOME, CATALINA\_HOME, and M2\_HOME (Maven) environment variables all set
* PATH environment variable has the JAVA bin and the Tomcat bin directories

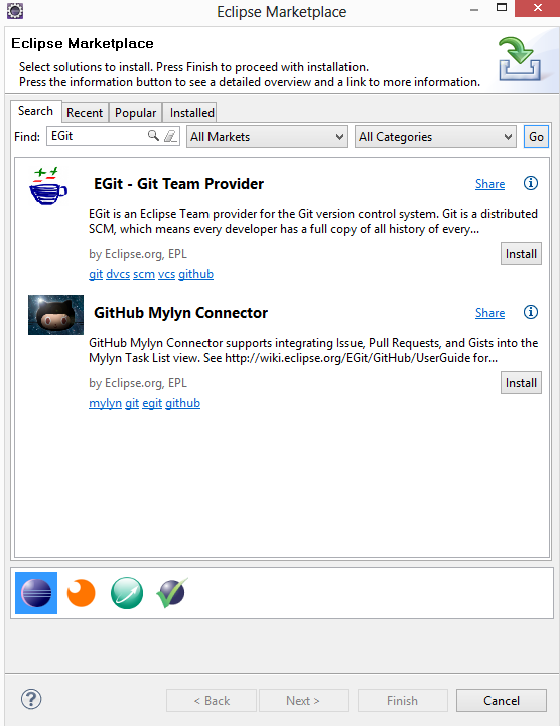
# Eclipse Plugins

Before retrieving the code from source control, a couple of Eclipse plugins must be installed and configured. The newest version of Eclipse includes a built in plugin search tool called Eclipse Marketplace. This guide assumes usage of that tool, so make sure you’re on Eclipse 3.7 or better.

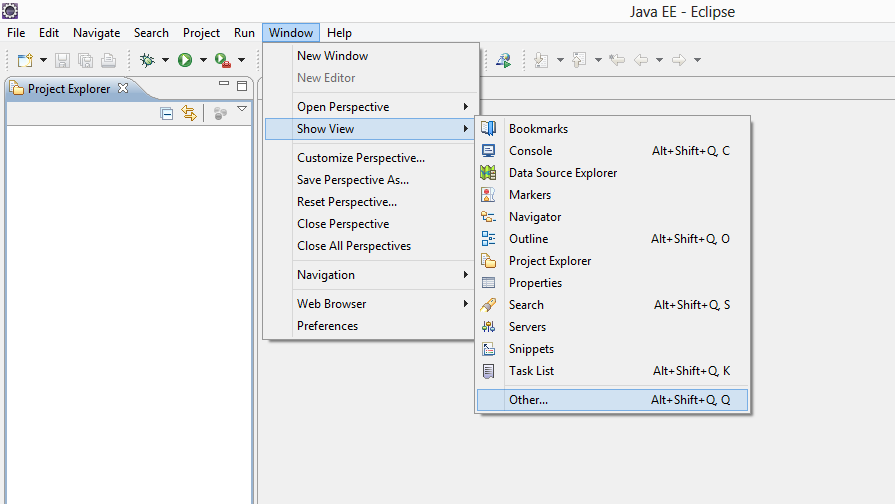


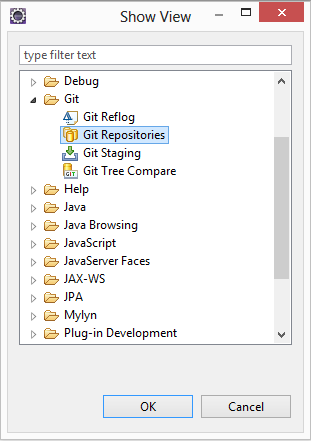
## EGit

The first plugin is for connecting to and retrieving source code from the Git repository. Open up the Eclipse Marketplace (found under Help) and search for “EGit”. Click on the install button and continue clicking through the prompts until it’s installed.

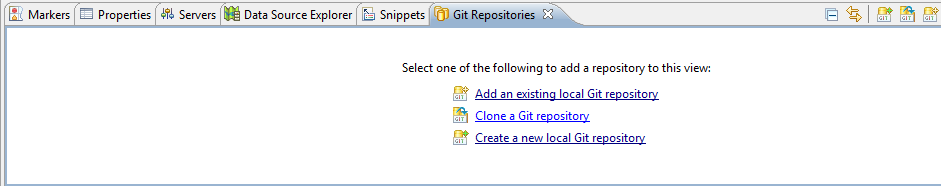


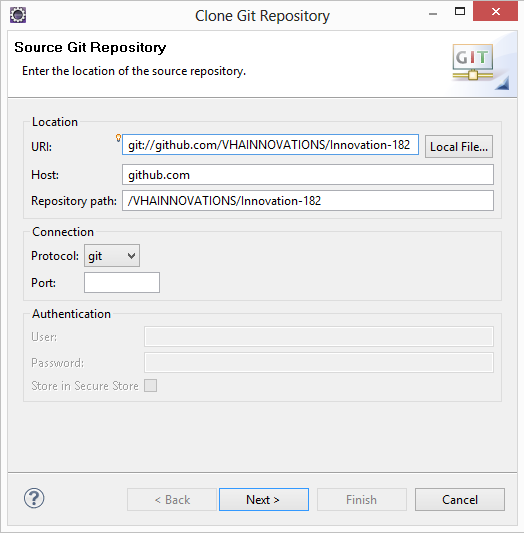
Once installed, it’s time to add the source repository. Under Window, go to Show View and then click on Other. In the Show View menu, expand the Git subcategory, select Git Repositories, and click OK. This will add a Git Repositories tab to the pane at the bottom of the main Eclipse window.

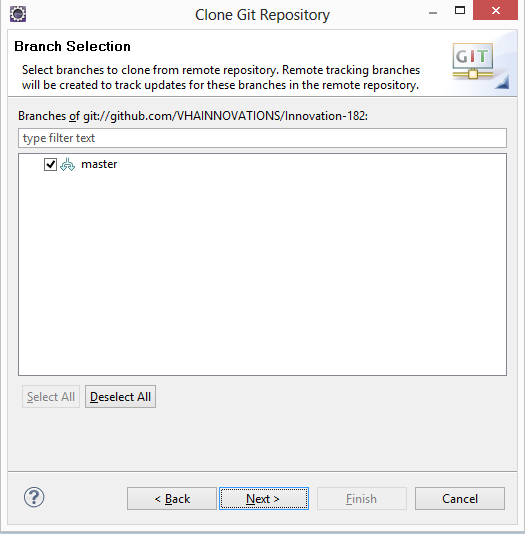


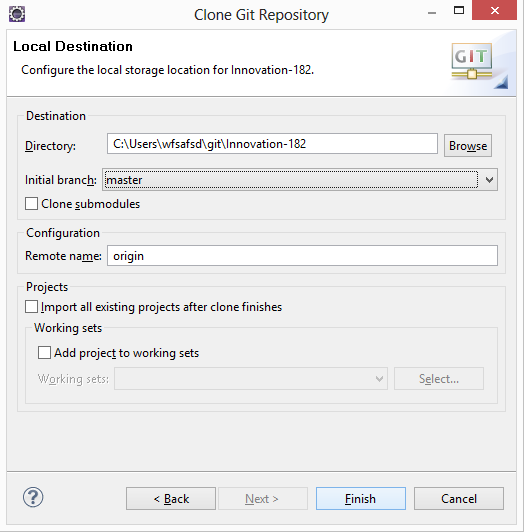


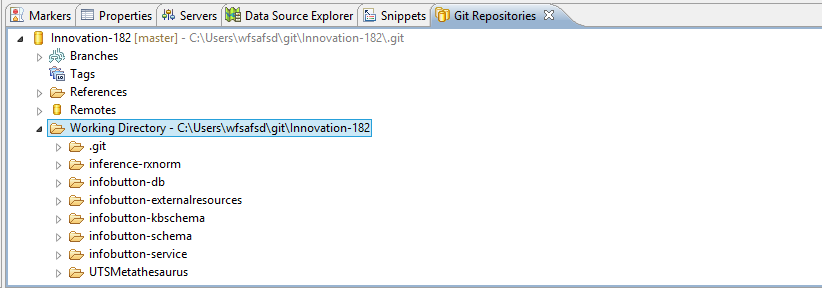
Select the Git Repositories tab, click on Clone a Git Repository. Enter git://github.com/VHAINNOVATIONS/Innovation-182 and click Next. Leave the default selection as Select All and click Next so that all the branches are cloned. Finally change your local destination directory if required and click Finish.







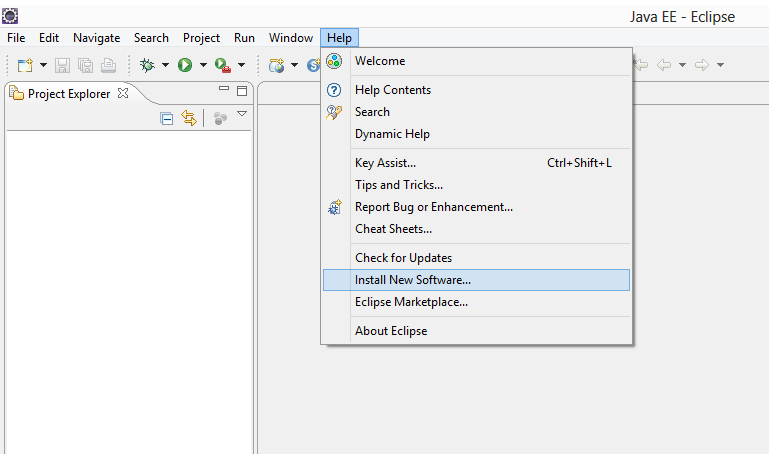


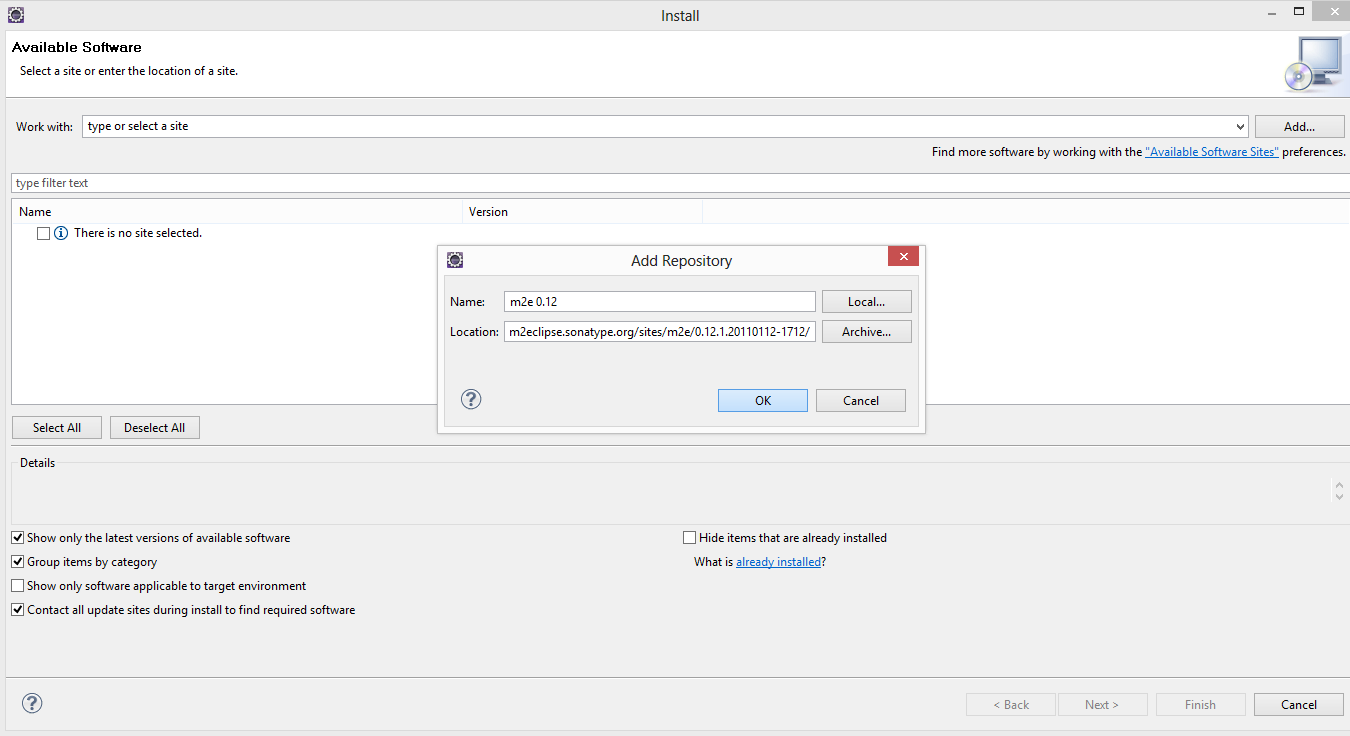


## M2E

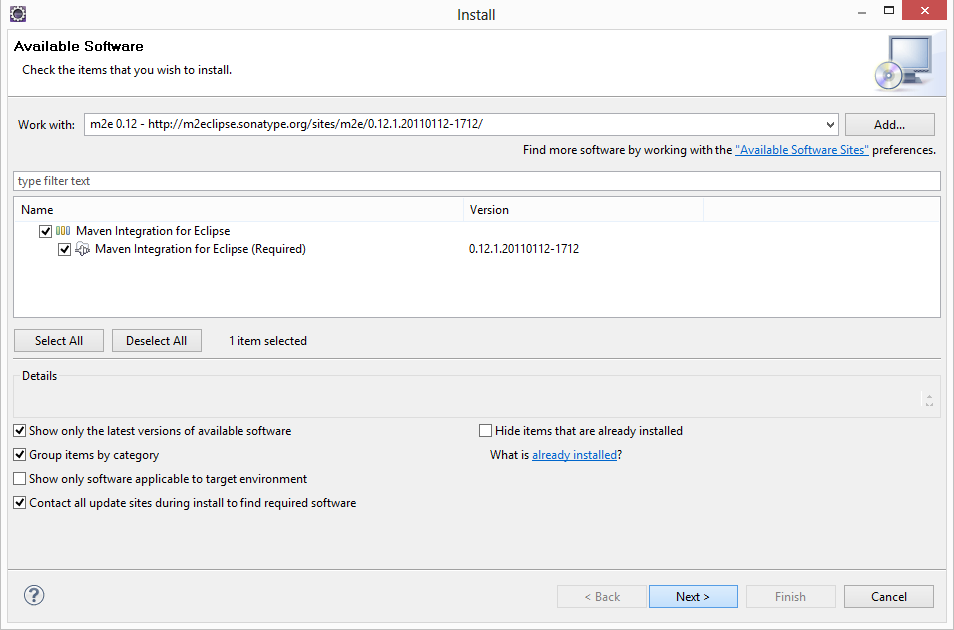
The next plugin is for dependency management and building with Maven. OpenInfobutton was developed using m2eclipse, which is now m2e. m2e 1.4 can be downloaded via Eclipse Marketplace or the Install New Software option on Eclipse.

* Market Place option: search for “m2eclipse”, click “Install” next to “Maven Integration for Eclipse WTP (Juno)”.
* Install New Software option: Click Help-Install New Software (screen shot below). Then click Add, and then enter the following URL in the Location section of the Add Repository window: <http://download.eclipse.org/technology/m2e/releases> (you can enter anything for name, then click OK).





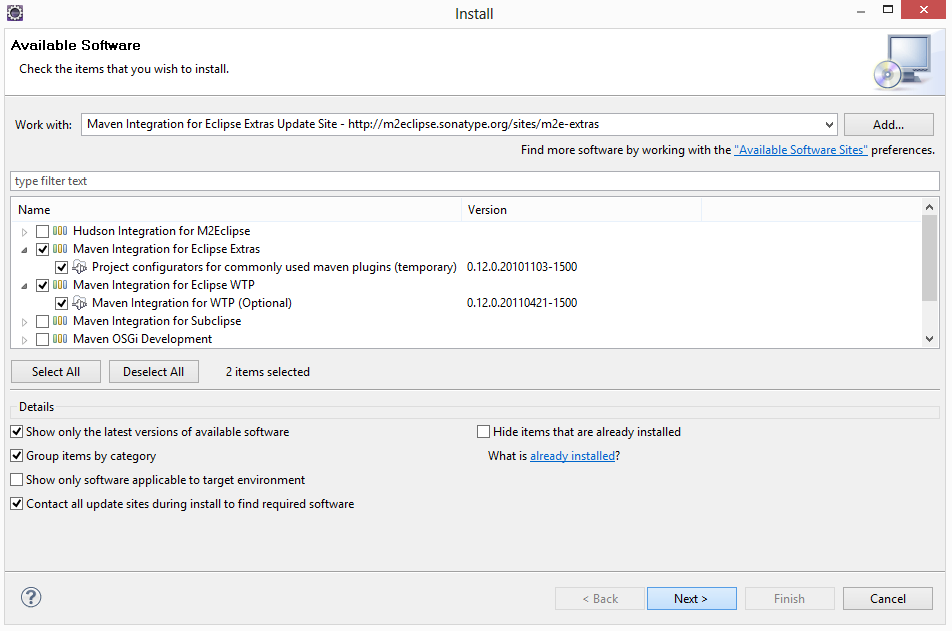
Now you should see a checkbox next to “Maven Integration for Eclipse”. Check it and click Next, Accept the license agreement, Finish to continue with installation. Restart Eclipse once installation is complete.



We also need to add a few m2e extras in the same way as m2e. This will enable Maven to pick up WebApps and make it easy to build within Eclipse. This time the URL to be added is <http://m2eclipse.sonatype.org/sites/m2e-extras> and name can be anything. Once this is added, eclipse will show you a list of checkboxes, select “Maven Integration for Eclipse Extras” and “Maven integration for Eclipse WTP.

**Important!** : Make sure that the versions are 0.12.

Follow through the series of prompts and complete the installation just like m2e.



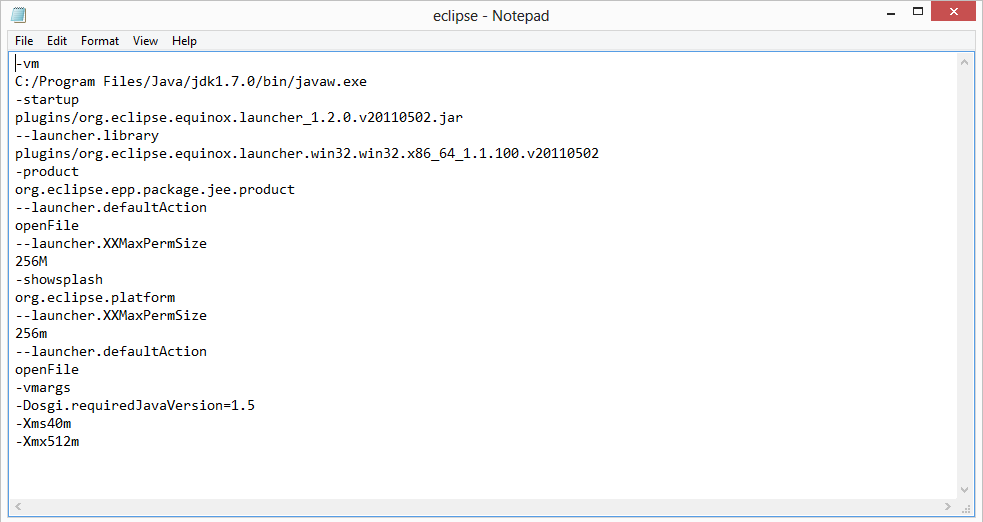
For Maven to work, we need to make sure that your Eclipse is running in a JDK.

On the Mac platform, this should already be configured for you. If you go to Eclipse > About Eclipse, then click on “Configuration”, you should be able to see if there is a “-vm” entry in the configuration details. If not, you will need to follow the steps for Windows, using the configuration file in {eclipse\_dir}\configuration\config.ini.

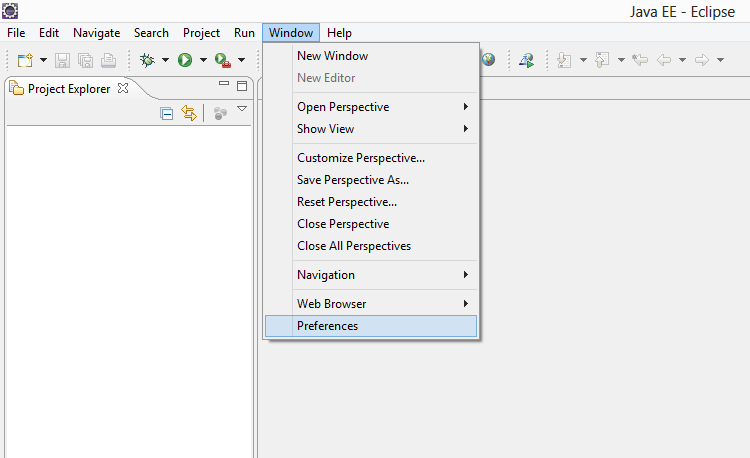
For Windows, go to your file system and navigate to your Eclipse directory. Should be something like C:\Program Files\eclipse. Open the file “eclipse.ini” in your preferred text editor, and add the follow lines to the top of the file,

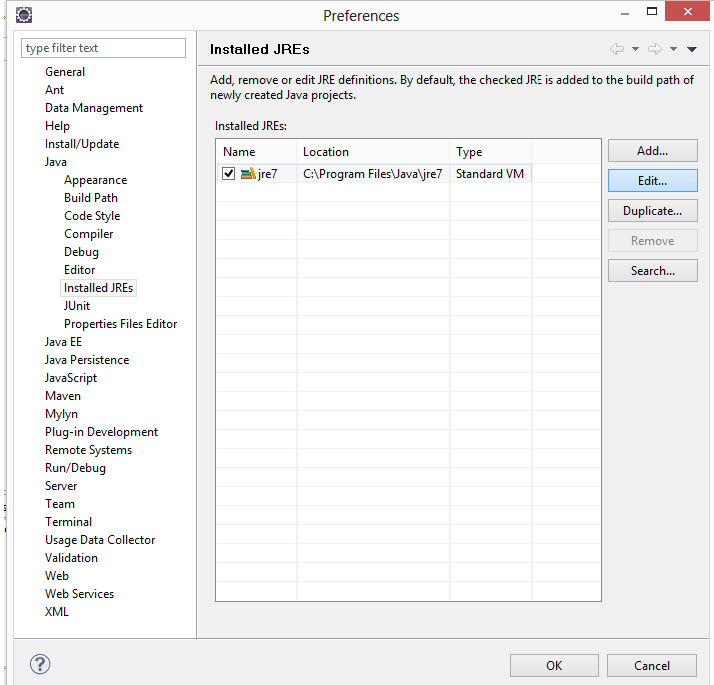
-vm

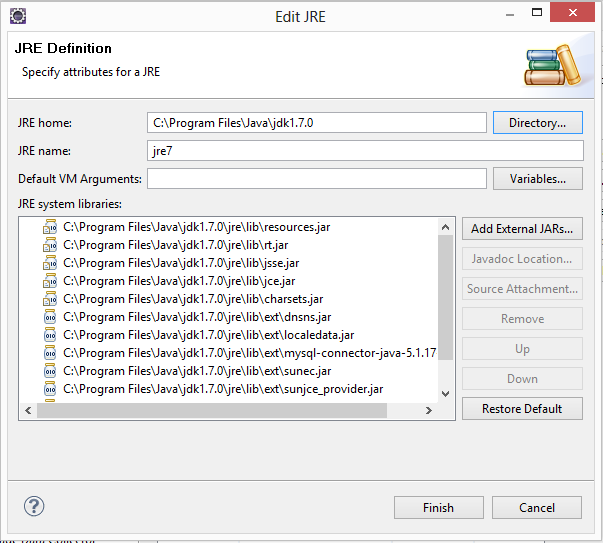
<path to your Java 6 JDK>\bin\javaw.exe



Make sure to restart Eclipse after doing this. You should also go to Window > Preferences > Java (on Mac, this is at Eclipse > Preferences > Java), and click on Installed JREs to check that default JRE is pointing to your Java 6 JDK rather than Eclipse’s internal JRE. If not, select it, click Edit, and make change the JRE home path to point to your JDK.







The final step for configuring Maven is creating settings.xml file. This file is for any global settings that apply to all your Maven projects. First, navigate to your Maven repository in your file system. If you’re using Windows XP, it should be located at,

C:\Documents and Settings\<username>\.m2 \

On Mac, it is in /Users/<username>/.m2/

Create a file in this directory called settings.xml and copy the following into it.

<settings xmlns="[http://maven.apache.org/SETTINGS/1.0.0](http://maven.apache.org/SETTINGS/1.0.0" \t "_blank)"

xmlns:xsi="[http://www.w3.org/2001/XMLSchema-instance](http://www.w3.org/2001/XMLSchema-instance" \t "_blank)"

xsi:schemaLocation="[http://maven.apache.org/SETTINGS/1.0.0](http://maven.apache.org/SETTINGS/1.0.0" \t "_blank)

[http://maven.apache.org/xsd/settings-1.0.0.xsd](http://maven.apache.org/xsd/settings-1.0.0.xsd" \t "_blank)">

<localRepository>C:\Documents and Settings\ai28\.m2\repository</localRepository>

<interactiveMode/>

<usePluginRegistry/>

<offline/>

<pluginGroups/>

<servers/>

</settings>

Make sure to edit the path in the <localRepository> element to reflect your personal file system. In most cases, all you should need to change is the user name. If you connect to the internet via a proxy, you also need to add the following element,

..

<proxies>

<proxy>

<active>true</active>

<protocol>http</protocol>

<host>[proxy.somewhere.com](http://proxy.somewhere.com/" \t "_blank)</host>

<port>8080</port>

<username>proxyuser</username>

<password>somepassword</password>

<nonProxyHosts>[www.google.com](http://www.google.com/" \t "_blank)|\*.[somewhere.com](http://somewhere.com/" \t "_blank)</nonProxyHosts>

</proxy>

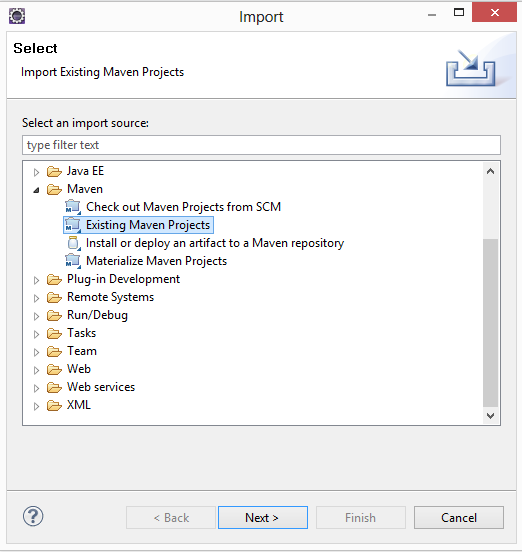
</proxies>

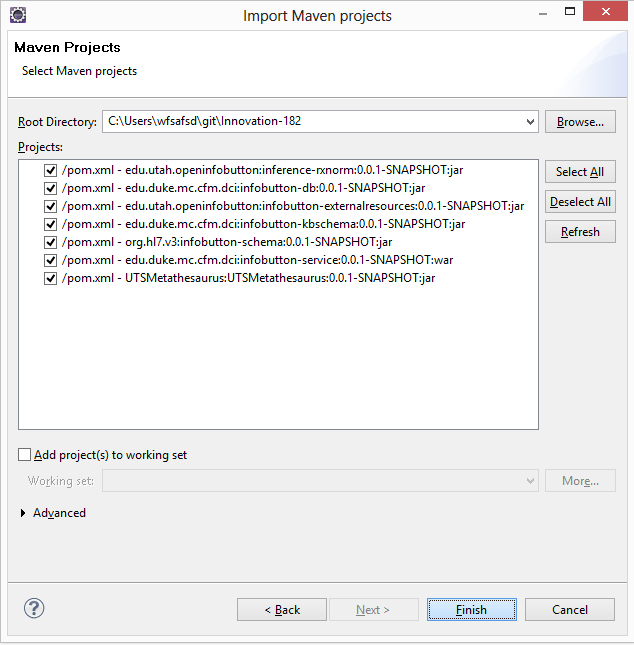
..

and fill in the correct information for your proxy server. You may need to restart Eclipse before any changes you make to the settings.xml file are reflected.

# Retrieval from GitHub

Since this project uses Maven, projects need to be imported in a very specific manner. When we did the setup for EGit, the project was cloned and it will be present on our file system. We just need to import those projects as Maven Projects. Go to File and click Import. Under Maven, select “Existing Maven Projects” and click Next. In the root directory, browse to the location where the project from GitHub was cloned (This location can be seen in the Git Repositories tab as well). Once this is done, 7 projects (pom files) will be identified. Leave them checked and click Finish and wait for the import to complete (this may take several minutes). The projects should now be visible in the project explorer.



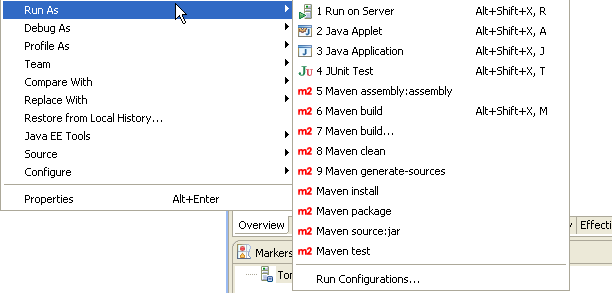


# Building

Infobutton-external resources depends on UTSMetathesaurus so right click the UTSMetathesaurus project and click on Maven and then on Update Project Configuration. This should generate the source code for the wsdls present in the pom and fix the dependency issue for infobutton-external resources.

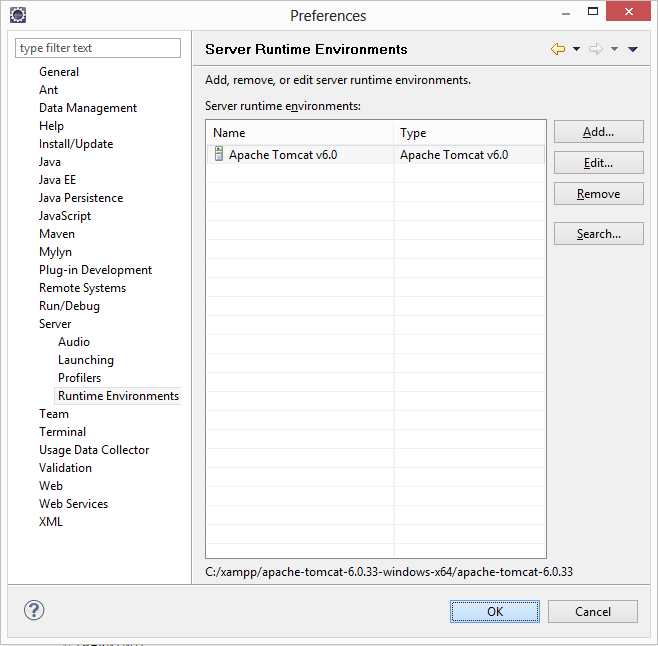
For building, infobutton-service includes the other five projects as dependencies. Therefore, they must be built first before attempting to build infobutton-service. Right click on each project and select Run As, then click on “Maven clean”. Wait for this to finish, then select Run As, Maven install. Repeat this for all the projects in the following order:

* Infobutton-kbschema
* Infobutton-schema
* Inference-rxnorm
* Infobutton-db
* UTSMetathesaurus
* Infobutton-externalresources
* Infobutton-Service



# Running

So now that the application is built and ready to run, it’s time to configure the Tomcat server. Go to Window and click on Preferences (on Mac this is Eclipse > Preferences). Expand the Server category and click on Runtime Environments. Click Add, select Apache Tomcat v6.0, and then click Next. Make sure the installation directory is pointing to your local Tomcat installation and then click Finish.



The final step is to setup your knowledge resource profiles and terminology inference profiles directory. Knowledge resource profiles consist of the OpenInfobutton knowledge base. These profiles contain instructions that configure the different resources that OpenInfobutton will access to. These profiles will be taken from the database. The terminology inferences will be accessed from the specified file system location.

To set the configuration details, navigate to infobutton-service/src/main/webapp/WEB-INF/serviceParams.properties. Here is where the various configurations details can be specified. Make sure you run the sql files in the database so that they are set up correctly. In case you are using UMLS, specify the username and password.

You might also want to edit infobutton-service/src/main/webapp/OpenInfobuttonDemo.html . In that file edit the baseURL inside the callIM function (which points to University of Utah) to your own host.

The infobutton-service project is a Dynamic web project and this needs to be picked up by eclipse. To do so, right click on the project (infobutton-service), and click Maven, then click Update Project configuration. Since we have the m2 extras, maven will pick the web.xml and the rest to make this a dynamic web project. Now we are all set.

Now the application can run. Go back to Eclipse, right click on infobutton-service in the Project Explorer and select Run As, Run on Server. Make sure to select the Tomcat server we configured earlier. Once running, try a test query.

